Graphical Update Caching for Mobile Thin Clients

Bert Vankeirsbilck

Supervisor(s): Bart Dhoedt, Filip De Turck

I. Introduction

Because of form factor constraints, required portability and battery lifetime, mobile devices have limited resources. Despite the rapid development of smaller and more efficient hardware, software requirements on CPU, memory and disk space cause significant hardware needs and battery drains. The general idea behind this article is to extend the mature thin client technology for use on mobile devices, offloading heavy computations to a distant server and leaving the applications unaltered. User input is sent over a network towards a server that executes the application, the generated screen updates are returned to the user's device. The functionality of the mobile device is reduced to presenting graphical output, capturing user events such as key strokes and pointer movements, and transmitting data over the network.

The overall goal of this doctoral study is to enable demanding applications on mobile thin clients. This will be reached by optimizing thin client computing concepts on the protocol level, as well as on the infrastructural level and the application level. The need for multi-layer optimizations is thoroughly explained in [1].

In this paper we propose graphical update caching as a method to reduce long term redundancies in thin client sessions. When analyzing the sequence of graphical updates generated by a typical user session on a desktop computer, we found that a lot of frames resemble others that have already been transmitted earlier in that session. Since we operate in a thin client environment where all graphical updates have to be sent over a wireless, limited bandwidth network, benefit can be found in this phenomenon by storing well-chosen key frames both at client and server side, and transmitting only the differences with respect to that frame. Since less data is to be received from the network at the client, the battery autonomy of the device decreases slower. This will contribute to the user satisfaction and will lessen the load on the environment.

We assess the bandwidth optimization potential of a static cache. This is a cache for graphical updates, that is filled before the thin client computing session starts. When the user logs in, the cache is loaded both at client and server side, and does not get altered during this session.

II. Statically choosing cache frames

We have taken a thin client usage session offline by storing the sequence of graphical updates and computed the byte-per-byte differences between subsequent full screen frames (in uncompressed format). We have taken the number of different bytes as a measure for the resemblance between frames. Through a matrix of mutual distances, we were able to identify the optimal combination of a predefined number of cache frames. Optimal cache frames are those that, combined, result in the smallest distance to the complete sequence. After visual inspection we found that the optimal cache frames represented the applications that were

B. Vankeirsbilck is with the Department of Information Technology of Ghent University (UGent), Belgium. E-mail: Bert.Vankeirsbilck@intec.UGent.be.

executing, i.e. the desktop background, the office program, the Google startup page and the homepage of the local newspaper.

III. Experimental results

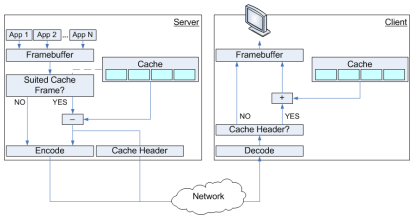

Figure 1. The caching architecture

The architecture for integrating a static cache in a thin client computing system is shown in Figure 1. At the server side, the executing applications of the user write their graphical output to the server's framebuffer. This framebuffer is analyzed in order to choose an optimal encoding method. If no suited cache frames are found for this graphical update, it will be directly encoded using a native encoding scheme from the used thin client protocol and sent to the client. If there is a well matching cache frame, this cache frame is subtracted from the graphical update and the difference is encoded using the classic encoding scheme and sent over to the client. In addition, a cache header containing necessary parameters such as the used cache frame has to be sent over to the client. At the client side, the received data is decoded, and depending on the presence of a cache header, the indicated cache frame is added to the decoded frame and delivered to the client framebuffer which eventually is presented on the screen of the client device. Figure 2 shows that optimally encoding with respect to one cache frame, the desktop background, already yields 20.56% bandwidth reduction. Adding an extra cache frame, the browser with the homepage of the local newspaper, brings the total bandwidth gain to 29.94%. Eventually, with five cache frames we achieved a bandwidth requirements decrease of 34.40% over classic encoding all updates.

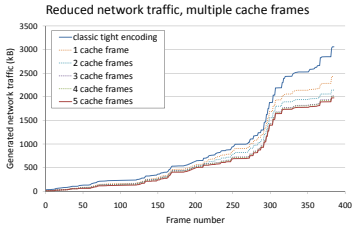

Figure 2. Influence of cache size on generated network traffic

The cache will be efficient in the case that a big update is requested. These big updates are expected when the user starts an application or when switching between applications, typically causing a full screen refresh. A video file, causing very fast successive screen refreshes will not be handled well by a cache though, since they will not often map to the cache frames that are to be predicted before the session starts. A solution for this, as suggested in [2], is to stream these video files.

IV. Conclusion

In thin client computing systems, benefit can be found in caching certain graphical updates. Other updates that resemble one of these cache frames can be efficiently coded by computing the difference to the cache frame, to be encoded and transported over the network to the thin client.

Acknowledgments

B. Vankeirsbilck is funded by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT).

References

[1] B. Vankeirsbilck, et al, *Bringing Thin Clients to the Mobile World*, 2008 NEM-SUMMIT, Towards Future Media Internet, Saint-Malo, France, 2008.

[2] P. Simoens, B. Vankeirsbilck, et al, *Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices*, Proceedings of Australasian Telecommunications and Network Applications Conference, Adelaide, Australia, 2008.